REMARKS

This response is to the Office Action mailed on 02/17/2011.

From the action:

This action is the non-final action in response to communication filed on 14 July 2010. Claims 18-28 are pending in the application. Claims 18-28 are rejected.

Applicant's response:

Acknowledged

From the action:

Claim Objections

Claim 20 is objected to because of the following informalities: Claim 20 recites "a test navigation template created for the purpose." It is unclear what limitation "created for the purpose" imposes (i.e. it does not appear to make grammatical sense) on the claim language and in general "created for" appears to be claiming a type of non-limiting intended use. The Examiner suggests the claim be amended such that the "created for a purpose" is removed from the claim. Appropriate correction is required.

Applicant's response:

Applicant herein amends claim 20 to overcome the objection.

From the action:

Claim Rejections - 35 USC § 103

Claims 18-28 are rejected under 35 U.S.C. 103(a) as being unpatentable over DaCosta et al (US-6,826,553 11/30/04) in view of Weinberg et al (US-6,360,332 03/19/02) in further view of Heninger (US-6,029,207 02/22/00) in further view of Bogrett (US-6,842,758 01/11/05).

-In regard to independent claim 18, DaCosta teaches a method for receiving automated notification of structural changes applied to electronic information pages accessed by a proxy network navigation and interaction system and effecting updates to navigation templates based on the change information, comprising steps of:

-establishing notification of failed execution (column 6, lines 9-13 & 35-41: "automatically requested"; column 18, lines 34-67: "script has failed... email or pager notification") of one of the navigation templates interacting with electronic information pages on a data-packet network (column 2, lines 11-35: "scripts...locates and extracts data" & 55-65: "specify and store a procedure"; column 3, lines 1-51: "structure...within the web page...recording a sequence of actions"; column 5, lines 36-55: "store navigation paths"; column 6, lines 51-67: "records navigation paths and associated steps"; column 7, lines 15-55: "navigation recording module captures each user generated event"; column 8, lines 19-21);

-recording/creating an instance of the failed navigation template associated with the cause of failure (column 18, lines 43-67: "it is known the script has failed...and proper notifications sent to individuals or entities responsible for the operation of the failing script by email...for example"; column 19, lines 1-15);

-accessing the notification of the of the failed navigation template for review purposes (column 6, lines 9-13 & 35-41; column 18, lines 34-67: i.e. developer accesses failed script for re-teaching purposes);

-being able to navigate to the electronic information page identified in the recorded instance (column 6, lines 9-13 & 35-41; column 18, lines 34-67: i.e. developer accesses failed script for re-teaching purposes);

-accessing/determining information necessary to repair the logic block involved in the failure (i.e. re-teaching a new navigation and extraction script by accessing the information).

-creating a new logic block according to the information and according to information contained in the recorded instance (column 6, lines 9-13 & 35-41; column 18, lines 34-67);

-installing the newly created logic block into the navigation template that failed, and into all existing navigation templates that depend on the failed logic block for successful function (column 6, lines 9-13 & 35-41; column 18, lines 34-67; column 19, lines 1-15).

DaCosta does not specifically teach wherein the instance of the failed navigation routine was stored for future review including parameters associated with the failed routine that included identification of at least a point of failure of at least one of a plurality of logic blocks used to build each of the navigation templates. Weinberg teaches storing the data file (column 2, lines 39-40; column 6, lines 19-22), wherein the application periodically submits test navigation and interaction routines (column 6, lines 19-22), and upon failure of the routine, creates a data file (column 2, lines 39-40; column 3, lines 29-43; column 6, lines 19-22; column 17, lines 10-52)(Fig. 5F), the data file comprising a point-of-failure indication within the failed routine and identifying the logic block of the template that failed (Fig. 5F: column 17, lines 17-21), parameters of the failure (column 17, lines 35-43), an identifier of the associated electronic page (columns 17-18: lines 62-l2)(Fig. 5F: "URL: www.mercint.com..), and stores the data file in the data repository sending notification of the action to the developer (column 2, lines 39-40; column 6, lines 15-23). It would have been obvious to one of ordinary skill in the art at the time of the invention to have stored the failed navigation script of DaCosta and for the proper notifications of the failed script to have included a point in process of the failure along with the identifier of the associated web page, because Weinberg teaches that by storing the failed navigation script, a developer can easily display the results of the navigation and quickly determine the location of the failure of the routine (column 3, lines 29-44). This would have made the re-teaching (i.e. correcting) of the navigation script in DaCosta easier for the developer (column 6, lines 9-13 & 35-41; column 18, lines 42-67).

DaCosta teaches wherein functional logic blocks were part of the navigation and interaction templates containing all of the possible navigation and interaction instructions required by the navigation system-interface module as defined by the a given user/developer (column 2, lines 20-31: "scripts...that locates and extracts data ...precisely locating and extracting the select data with a granularity specified by the user" & lines 57-67: "capability for a user to specify...in an automated manor"; column 5, lines 29-55: "learn and store navigation paths...dialogs and forms that need to be filled...login name and password"; column 7, lines 16-28: "captures each user-generated event."; columns 7-8, lines 55-5: "automatically repeatedly query a web site...upon a single exemplematic query"; column 9, lines 5-44). Neither DaCosta nor Weinberg specifically teach wherein the functional logic blocks in the defined interaction scripts were modular parts of the interaction scripts. Heninger teaches building software components in a modular fashion such that each modular component could be constructed, modified, and tested independently (column 1, lines 20-29). It would have been obvious to one of ordinary skill in the art at the time of the invention for the functional logic blocks of DaCosta to have been modular parts of the navigation and interaction templates, because Heninger taught that computer software developers realize that modular interacting software components provide the advantages of being more easily designed, generated, tested, installed, and maintained as well leading to better computer products at a minimal cost (column 1, lines 20-67; column 2, lines 1-24). Thus the modular software components of Heninger would have provided the developers of DaCosta a better way of maintaining, editing, and correcting failed navigation scripts (column 18, lines 34-67) by allowing the developers to fix only the modular part of the failed navigation and interaction script.

The modified DaCosta reference teaches wherein the functional site-logic blocks of the navigation and interaction templates could be modular parts of the interaction scripts and replacing the defunct site logic in a modular fashion. The modified DaCosta does not specifically teach automatically replacing/installing only the defunct site-logic block with one or more operational site-logic blocks. Bogrett teaches method for

maintaining client applications wherein all upgrades to client software was done automatically by a server by generating an incremental update for transmitting to the client, wherein the incremental update includes only the modules of code that need to be updated (column 10, lines 25-67: "upgrades to the client software are done automatically by the server...generates an incremental update...transmits the incremental update to the client....modules...are out-of-date, missing, or obsolete and then selectively pushes the correct modules")(Fig. 4). It would have been obvious to one of ordinary skill in the art at the time of the invention for the software updating of the modified DaCosta to have included only updating the defunct software modules as taught in Bogrett, because Bogrett taught that by only updating the defunct software modules in an incremental update the client gained the benefit of executing quickly and always being up-to-date and the system gained the benefit of minimizing network traffic (column 10, line 65-column 11, line 2: "never have to manually update...executes quickly and is always up-to-date...network traffic is minimized").

-In regard to dependent claim 19, DaCosta teaches wherein the data-packet network could be the Internet (column 2, line 13: "Internet") and wherein the electronic information page was a web page (column 2, line 13: "web site") hosted on the network.

-In regard to dependent claim 20, DaCosta teaches wherein the navigation routine was performed according to a test navigation template (column 6, lines 9-13 & 35-41; column 18, lines 54-65)(Fig. 2: i.e. according to the navigation and extraction scripts).

-In regard to dependent claim 21, DaCosta teaches wherein the navigation routine was performed according to a client navigation template (Fig. 7: "User").

-In regard to dependent claim 22, the modified DaCosta teaches wherein the recorded instance of the failed routine was created in the form of a data file and stored in a data repository accessible through a network (column 18, lines 54-67).

-In regard to dependent claim 23, DaCosta teaches wherein the recorded instance of the failed navigation routine was accessed by a software developer (column 6, lines 9-13 & 35-41; column 18, lines 54-67).

-In regard to dependent claim 24, DaCosta teaches wherein navigation was performed by the developer utilizing an instance of a browser installed on a computerized workstation (column 2, lines 11-30: "browser").

-In regard to dependent claim 25, the modified DaCosta teaches wherein the new logic was in the form of a modular logic block installable to a navigation template (column 6, lines 9-13 & 35-41; column 7, lines 18-54: "programmatically modify the recorded path"; column 18, lines 54-67).

-In regard to dependent claim 26, the modified DaCosta teaches wherein the new logic block self-installs to a depended navigation template (column 6, lines 9-13 & 35-41; column 18, lines 42-67: "ensure each of the users has a corrected script as soon as possible, i.e., as soon as it is downloaded to the central repository...running the script").

-In regard to dependent claim 27, the modified DaCosta teaches testing the new logic before the implementation (column 19, lines 1-15: "determine whether it is operating correctly").

-In regard to dependent claim 28, the modified DaCosta teaches creating more than one logic block within a navigation template (column 2, lines 11-35: "scripts...locates and extracts data" & 55-65: "specify and store a procedure"; column 3,

lines 1-51: "structure... within the web page...recording a sequence of actions"; column 5, lines 36-55: "store navigation paths"; column 6, lines 51-67: "records navigation paths and associated steps"; column 7, lines 15-55: "navigation recording module captures each user generated event"; column 8, lines 19-21) and wherein more than one block could fail (column 6, lines 9-13 & 35-41; column 18, lines 34-67; column 19, lines 1-15).

Applicant's response:

Applicant herein amends claim 18 to specifically recite that only parameters of the navigation template associated with the cause of failure during live execution are recorded and automatically identifying the defunct site logic block in all existing stored navigation templates that depend on the failed logic block and replacing it with the newly created modular logic block. Applicant's claim 18, as amended, is reproduced, below:

18. A method for receiving notification of structural changes applied to electronic information pages accessed by a proxy network navigation and interaction system and effecting updates to navigation templates based on the change information, comprising steps of:

(a) establishing notification of failed execution of one of the navigation templates interacting with electronic information pages on a data-packet-network, the navigation templates assembled from a plurality of functional logic blocks;

(b) recording only parameters of the navigation template associated with the cause of failure during live execution at the electronic information pages by the proxy, including identification of at least one modular logic block involved at the point of failure;

(c) accessing the recorded instance of the failed navigation template for review purposes;

(d) navigating to the electronic information page identified in the recorded instance;

(e) determining information necessary to repair the logic block involved at the point of failure;

(f) creating a new modular logic block according to the information; and

(g) automatically installing the newly created modular logic block into the navigation template that failed, and automatically identifying the defunct site logic block in all existing stored navigation templates that depend on the failed logic block and replacing it with the newly created modular logic block.

The Examiner states DaCosta teaches, "-recording/creating an instance of the failed navigation template associated with the cause of failure (column 18, lines 43-67: "it is known the script has failed...and proper notifications sent to individuals or entities responsible for the operation of the failing script by email...for example"; column 19, lines 1-15);" Applicant's limitation, as amended, teaches:

(b) recording only parameters of the navigation template associated with the cause of failure during live execution at the electronic information pages by the proxy, including identification of at least one modular logic block involved at the point of failure;

Applicant points out that DaCosta teaches that the script can then be automatically disabled, and proper notifications sent to individuals or entities responsible for the operation of the failing script by e-mail or pager notification, for example. Applicant argues that DaCosta does not teach recording the failure while executing the navigation template; **DaCosta teaches emailing a notification** of a failure which is not the claimed limitation.

The Examiner states DaCosta teaches, "-accessing the notification of the of the failed navigation template for review purposes (column 6, lines 9-13 & 35-41; column 18, lines 34-67: i.e. developer accesses failed script for re-teaching purposes);" Applicant's claim recites:

(c) accessing the recorded instance of the failed navigation template for review purposes;

Applicant points out that the Examiner errs when re-wording applicant's claim limitations in order to apply the art. This is inappropriate examination procedure. The art of DaCosta clearly fails to teach recording and storing the parameters of the navigation routine involved in the failure, as claimed.

The Examiner states DaCosta fails to teach wherein the instance of the failed navigation routine was stored for future review including parameters associated with the failed routine that included identification of at least a point of failure of at least one of a plurality of logic blocks used to build each of the navigation templates. Weinberg teaches storing the data file (column 2, lines 39-40; column 6, lines 19-22), wherein the application periodically submits test navigation and interaction routines, and upon failure of the routine, creates a data file, the data file comprising a point-of-failure indication within the failed routine and identifying the logic block of the template that failed, parameters of the failure, an identifier of the associated electronic page, and stores the data file in the data repository sending notification of the action to the developer.

Applicant argues that the Examiner has effectively side-stepped the requirement to show art of recording the failure in the navigation template, as claimed. Applicant argues Weinburg teaches storing a Web page and submitting it to a testing facility in order to test performance of a server actions and response to commands while performing the navigation. Weinburg teaches; "The execution summary includes a tree representation 84 or "report tree" of the test execution in the left pane of the screen. Each iteration of the test and the associated steps are presented as nodes of the tree 84." Applicant argues that Weinburg's testing facility creates a new electronic document representing the Web page in the form of a tree of steps in order to graphically pinpoint step failures of the server responses. Weinburg fails to teach recording and storing of the actual failure of the navigation template during actual navigation on the network, as claimed. Applicant believes that the combination of DaCosta and Weinburg fail to teach recording and storing the failure, as claimed.

The Examiner states, " It would have been obvious to one of ordinary skill in the art at the time of the invention to have stored the failed navigation script of DaCosta and for the proper notifications of the failed script to have included a point in process of the failure along with an identifier of the associated web page, because Weinberg teaches that by storing the failed navigation script, a developer can easily display the results of the navigation and quickly determine the location of the failure of the routine (column 3, lines 29-44). This would have made the re-teaching (i.e. correcting) of the navigation script in DaCosta easier for the developer (column 6, lines 9-13 & 35-41; column 18, lines 42-67)." Applicant argues that pinpointing a decision step in a decision tree in a document (not the Web page or navigation script) created to test responses of a server, as in Weinburg, would not enable a developer to easily display the results of the navigation and quickly determine the location of the failure of the routine. Applicant points out that DaCosta teaches recording portions of an actual navigation on a Web page, not the server routine of Weinburg. An error in the decision tree document of Weinburg could not easily be related to pinpoint a defunct module identified in the navigation scripts of DaCosta. There is additional steps that would have to be taken that would not be apparent to one with skill in the art. Therefore, the combination would not provide a benefit to the skilled person looking to achieve detecting, recording and repairing navigation scripts during live navigation on the network.

The Examiner states, " Neither DaCosta nor Weinberg specifically teach wherein the functional logic blocks in the defined interaction scripts were modular parts of the interaction scripts. Heninger teaches building software components in a modular fashion such that each modular component could be constructed, modified, and tested independently. Heninger would have provided the developers of DaCosta a better way of maintaining, editing, and correcting failed navigation scripts (column 18, lines 34-67) by allowing the developers to fix only the modular part of the failed navigation and interaction script." Applicant points out that the navigation scripts of DaCosta are recordings of a users navigation at a Web site. Applicant argues that the combination would still lack the ability to associate the software blocks of Heninger with the recorded

portions of navigation scripts. One with skill in the art would easily jump to associating recording with modular software. Again there are additional steps one with skill would have to take to achieve the benefit of the combination in relation to applicant's claimed invention that are not presented in the art, as required to produce a benefit of the combination.

Applicant believes that claim 18, as amended, is patentable over the combination of the art presented by the Examiner for at least the arguments presented above. Claims 19-24 and 26-28 are patentable on their own merits, as amended, or at least as depended from a patentable claim. Claim 25 is herein cancelled.

Summary

As all of the claims, as amended and argued above, have been shown to be patentable over the art presented by the Examiner, applicant respectfully requests reconsideration and the case be passed quickly to issue.

If any fees are due beyond fees paid with this amendment, authorization is made to deduct those fees from deposit account 50-0534. If any time extension is needed beyond any extension requested with this amendment, such extension is hereby requested.

Respectfully Submitted
Tim Armandpour et al.

By */Donald R. Boys/*
Donald R. Boys
Reg. No. 35,074

Central Coast Patent Agency, Inc.
3 Hangar Way, Suite D
Watsonville, CA 95076
(831) 768-1755